Project Report: Digit Recognizer

Jimmy Lin (xl5224)
Department of Computer Science
University of Texas at Austin
Austin, TX 78712
JimmyLin@utexas.edu

Jisong Yang (jy7226)
Department of Statistical Scientfic Computation
University of Texas at Austin
Austin, TX 78712
jsyang993@gmail.com

Abstract

BACKGROUND. The idea of recognizing digit . The goal in this project is to take an image of a handwritten single digit, and determine what that digit is. Kaggle competition. We propose to

1 Introduction

Recognizing hand-written characters has long been a hot topic in artificial intelligence community. Handwriting recognition is the ability of a computer to receive and interpret intelligible handwritten input from sources such as paper documents, photographs, touch-screens and other devices. In the past decades, many probabilistic or non-probabilistic algorithms have been exploited to effectively solve this problem, including K-nearest neighbour, random forest, and well-known neural network.

In this report, we will illuminate the motivation of digit recognition problem on section 2. In section 3, the main algorithms exploited in this project will be present to reader. For each algorithm, the algorithmic introduction, the reason of attempting that algorithm and finally corresponding implementation details will be indicated.

2 Motivations

As to the historical application of handwriting recognition, it can be traced back to early 1980s. There are plenty of commercial products incorporating handwriting recognition as a replacement for keyboard input were introduced. The hardware application continued to develop in the following decades. Until the recent years, Tablet PCs can be regarded as a special notebook computer that is outfitted with a digitizer tablet and a stylus, and allows a user to handwrite text on the unit's screen. In addition, highly efficient handwriting recognizers were developed in real world to apply on the zip codes detection.

Plenty of mature solutions has already been developed to existence in solving the handwriting problem. Since 2009, the recurrent neural networks and deep feedforward neural networks developed in the research group of Jrgen Schmidhuber at the Swiss AI Lab IDSIA have outshined many other models in competitions. The fact that existing algorithms are powerful in recognizing digits does not remove possibility of further improvement. The chance of improvement lies in reduction of computational complexity for recognizing digits in higher degree.

3 Implementation Details

3.1 Data Preprocessing

Principal Component Analysis (PCA) is a statistical procedure that uses orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly

uncorrelated variables called principal components. Typically, the number of principal components is no more than the number of original variables. That is to say, a particular subspace, whose bases are independent to each other, is generated by such tranformation. This transformation is defined in such a way that the first principal component has the largest possible variance, and each succeeding component in turn has the highest variance possible under the constraint that it is orthogonal to the preceding components. In section 4, the performance comparisons of different algorithms will be illustrated and after which, we will make conclusion about which one achieve the best performance.

In our project, we make use of PCA as dimensionality reduction for preprocessing the raw input image data. Since the raw data explictly characterize the gray value of each pixels, each handwritten instance is extremely high-dimensional. Although directly using the raw input data avoids loss of information, the computational complexity would be unresonably large. Put it another way, we have to think of a way to compromise the ultimate accuracy to achieve a less computational cost on that algorithm.

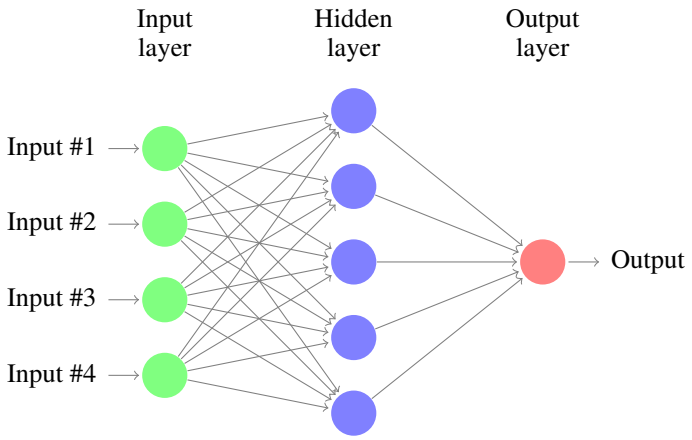
However, several details matter significantly to the effect of PCA. One is the number of principal components employed. If too many principal components are used, it is far from reaching the goal of dimensionality reduction. However, we may lose useful information for latter prediction if important components are not used. Note that even least useful principal components contain disjoint information that may contribute to latter prediction performance. As to our implementation, we choose to pick up the set of principal components whose eigenvalues are significantly greater than the others.

[TODO: How many components do we use in total?]

3.2 Neural Network

In the field of machine learning, **neural networks** are computational models inspired by animals' central nervous systems (in particular the brain) that are capable of machine learning and pattern recognition. They are usually presented as systems of interconnected "neurons" that can compute values from inputs by feeding information through the network.

The graphical representation of variables are as follows. The nodes in input layer represents the input features of each training entity. Those nodes in hidden layer corresponds to the learned features, served for final regression or classification decision of that entity. And the only node in output layer corresponds to the regression or prediction decision.



[TODO: brief technical introduction and notation]

Mathematically,

To train neural network model, we need to feed the neural network with a series of entities. The flow of information, while doing training, involves in two stages: forward propagation and backward propagation. In the stage of forward propagation, the network accept a new training entity at input layer , pass the value forward until output layer and finally evaluate the prediction over that particular entity. Since the error (difference between prediction label and groudtruth label) was derived, it

comes to the stage of the back propagation. In this stage, the pre-computed error was passed back towards the input layer so as to update the parameters (weight matrix). We can keep on feeding training entity to this neural network until convergence.

As to the motivation of using neural network, one most significant factor comes to the fact that Neural Networks are usually employed as one universal function approximator. Formally, if given sufficent hidden nodes, the neural network model can approximate any function to arbitrary precision. Ituitively, we can treat the digit number as a function of a series features. In original setting, these features are gray value of every pixel. If preprocessed by PCA, the features become the principal components chosen by us.

There are also several concerns for the implementation of Neural Network model. The first one coming to our mind is the number of hidden nodes for usage. Another concern is the determination of leraning rate α. The third one is selection of activation function for hidden nodes. Typically, there are a few options: $tan(x)$, $log(x)$, $exp(x)$. [TODO: further expansion]

3.3 Support Vector Machine

3.4 Deep Learning

4 Result Comparison

5 Conclusion

6 References

[1] Liu, C. L., Nakashima, K., Sako, H., & Fujisawa, H. (2003). Handwritten digit recognition: benchmarking of state-of-the-art techniques. *Pattern Recognition*, **36(10)**, 2271-2285.

[2] Le Cun, B. B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., & Jackel, L. D. (1990). Handwritten digit recognition with a back-propagation network. In *Advances in neural information processing systems.*

[3] LeCun, Y., Jackel, L. D., Bottou, L., Brunot, A., Cortes, C., Denker, J. S., & Vapnik, V. (1995, October). Comparison of learning algorithms for handwritten digit recognition. In *International conference on artificial neural networks* **(Vol. 60)**.

[4] Hinton, G. E., Osindero, S., & Teh, Y. W. (2006). A fast learning algorithm for deep belief nets. *Neural computation*, **18(7)**, 1527-1554.

[5] Ciresan, D. C., Meier, U., Gambardella, L. M., & Schmidhuber, J. (2010). Deep, big, simple neural nets for handwritten digit recognition. *Neural computation*, *22(12)*, 3207-3220.

A PCA

B Neural Network

C Multi-class Support Vector Machine

D Deep Learning